Improving Information Flow in ReLU Neural Networks: Weight Matrix Problems and Possible Fixes

1. Introduction

Rectified Linear Units (ReLUs) are widely used activation functions in modern deep neural networks. They are simple, fast, and effective at reducing vanishing gradients. However, they introduce a structural problem in weight matrices that can restrict information flow and diminish modelling power.

This report explores how ReLU-induced *row and column zeroing* limits the effective use of weights in a network, and how alternative strategies such as the **Fast Walsh-Hadamard Transform (FWHT)** or **Concatenated ReLU (CReLU)** can help.

We will also discuss how these approaches compare to traditional solutions like **residual connections** and offer additional insights on designing networks for better information propagation.

---

2. The Row and Column Zeroing Problem

2.1 Row Zeroing

When a neuron's output passes through ReLU and becomes zero, its contribution to the next layer disappears.

- The **row** in the current layer's weight matrix corresponding to that neuron is effectively *zeroed out*.
- The weights in that row exist, but they no longer influence the computation for that input.

This means that valuable parameter capacity is wasted whenever ReLU suppresses a neuron.

2.2 Column Zeroing

The problem worsens at the next layer:

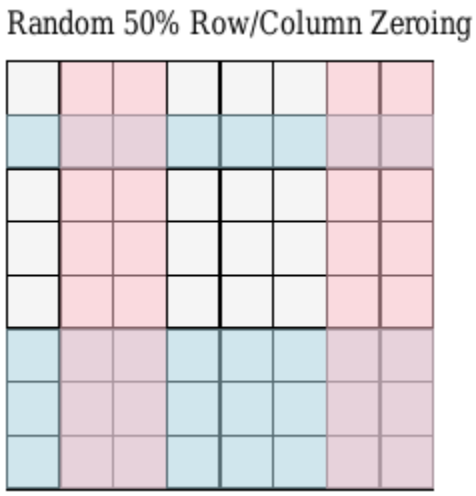
- The zero output from the current layer becomes part of the input vector for the next layer.
- This forces the **entire column** in the next layer's weight matrix to multiply by zero.
- Thus, that column's weights are also "lost" to computation for that input.

2.3 Compounding Effect

If ReLU activations produce zero outputs for ~50% of neurons (a common scenario with symmetric input distributions), then:

- 50% of rows are inactive, and
- 50% of columns in the next layer are inactive.

Together, only about **25% of the network's weights** are active for any given input. This is a dramatic under-utilization of model capacity.



---

3. The Fast Walsh-Hadamard Transform (FWHT) Fix

3.1 One-to-All Connectivity

The **Fast Walsh-Hadamard Transform (FWHT)** is a structured orthogonal transform that redistributes input information across all output channels without introducing additional learned parameters.

- If only **one neuron** has a nonzero output, the FWHT spreads that signal across *all outputs*.

- As a result, the risk of entire columns being zeroed out in the next weight matrix is drastically reduced.

3.2 Advantages

- **Prevents column zeroing**: Every column receives nonzero input, preserving weight usage.

- **No new weights**: The FWHT is a fixed transform, not a learned one.

- **Computationally efficient**: The FWHT runs in *O(n log n)* time, similar to the FFT, and can be hardware-optimized.
- **Self-Inverse:** As typically implemented the FWHT is self-inverse meaning the backpropagation inverse is simply to run the error signal through the FWHT.

---

4. The Concatenated ReLU (CReLU) Fix

4.1 Hidden Weight Twins

CReLU works by duplicating each neuron into two parallel channels: one that responds to positive inputs, another to negative inputs.

- Each column in the next layer's weight matrix has a **hidden twin**.

- Depending on the sign of the pre-activation value, one twin becomes active while the other remains hidden.

4.2 Advantages

- **Preserves information about sign**: Unlike standard ReLU, which discards negative signals, CReLU retains both polarities.

- **Mitigates weight loss**: Since one of the twins is always active, fewer weights are lost due to ReLU's zeroing effect.

- **Increased modelling flexibility**: The nonlinearity is shifted into a discrete *choice* of which weight twin to use, improving representational power.

4.3 Trade-Offs

- Requires **doubling the number of weights**, since every neuron is paired.
- Half of these weights are inactive at any one time, but overall coverage of information flow is improved.

---

5. Comparison with Residual Networks

Residual networks (ResNets) address information loss by adding **skip connections** that allow signals to bypass nonlinear transformations.

- This ensures some gradient and input flow persists even if intermediate layers suppress most activity.
- However, skip connections introduce structural complexity and additional computation.

By contrast:

- FWHT prevents weight starvation *within the existing network* by spreading signals.
- CReLU preserves polarity information without needing explicit shortcuts.

Both methods may **reduce reliance on ResNet-style designs** for maintaining information flow, offering simpler alternatives.

---

6. Insights and Further Considerations

1. **Weight Efficiency Matters**: ReLU-induced weight loss means networks are often larger than necessary just to ensure enough active capacity. Mitigations like FWHT or CReLU can make smaller networks more efficient.

2. **Structured Transforms as Bridges**: Orthogonal transforms (FWHT, FFT, random projections) can serve as lightweight bridges between layers, improving mixing without adding parameters.

3. **Activation Alternatives**: Other nonlinearities (e.g., Leaky ReLU, GELU, ELU) partially address the zeroing issue but do not solve column starvation as systematically as FWHT.

4. **Dynamic Synthetic Weights**: Viewing inactive weights as "hidden" rather than "lost" suggests a reinterpretation of ReLU networks as **dynamic synthetic weight matrices**—where weight usage depends on input-dependent activation masks. FWHT and CReLU make these synthetic matrices more expressive.

---

7. Conclusion

ReLU neural networks suffer from an under-appreciated structural flaw: row and column zeroing that drastically reduces the number of active weights per input. This weakens information flow and modelling capacity.

Two effective fixes are:

- **FWHT insertion**, which spreads surviving signals across all channels, preventing column starvation without adding parameters.

- **CReLU activation**, which duplicates neurons to preserve polarity information and ensure fewer weights are wasted.

Both approaches can enhance information flow and reduce the need for heavy architectural interventions like residual connections. Together, they highlight an important design principle: **effective use of existing weights matters as much as adding more weights**.